Some more quantum building blocks and the Deutsch-Jozsa algorithm

Dr. Maria (Masha) Okounkova
mokounkova@flatironinstitute.org

Reading references:

- Aaronson, Quantum Information Notes, Lecture 17
- Nielsen and Chuang, Quantum Computation and Quantum Information, Sec. 1.4

Two minute essay:

1. What did I learn from the reading that I didn't know before?
2. What is confusing?
3. What do I want to know more about?

Contents

1 Why we care: quantum vs. classical complexity

The purpose of today's class is to appreciate just how powerful quantum computing is over classical computing. When studying quantum algorithms, we can compute their *complexity*, and compare this to that of classical algorithms.

For example, for an unstructured search over N inputs (these could be, for example database entries), the classical algorithm takes $\mathcal{O}(N)$ search queries, while the quantum case, using Grover's algorithm, takes $\mathcal{O}(\sqrt{N})$ search queries.

A more popular quantum computing example is finding the prime factorization of an N-bit integer. The best-known classical algorithm, the general number-field sieve, takes $\mathcal{O}(e^{N^{1/3}})$ evaluations. The fact that this is a large number of evaluations, and hence factoring is 'hard', is the basis of many modern cryptography systems. In the quantum case, using Shor's algorithm, factoring would take $\mathcal{O}(N^3)$ evaluations.

There are many examples of algorithms where we can compare their quantum and classical complexity, and the Quantum Algorithm Zoo is a great resource to explore.

Today we'll learn some quantum computing building blocks, and build to the Deutsch-Jozsa algorithm, which solves a problem that in the classical case requires $\mathcal{O}(2^{N-1})$ evaluations with just $\mathcal{O}(1)$ evaluations in the quantum case.

2 Building block: qubits (Review)

Last time together, we talked about qubits. For a quick review:

2.1 One qubit

A qubit is a two-level quantum system, with an amplitude for 0 and 1. We will use the notation

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \text{ and } |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{1}$$

and can write a qubit in a state

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{2}$$

where α and β are complex amplitudes, the probability of measuring $|0\rangle$ is $\|\alpha\|^2$, and the probability of measuring $|1\rangle$ is $\|\beta\|^2$.

Physical realizations of two-level quantum systems that can be used as qubits include

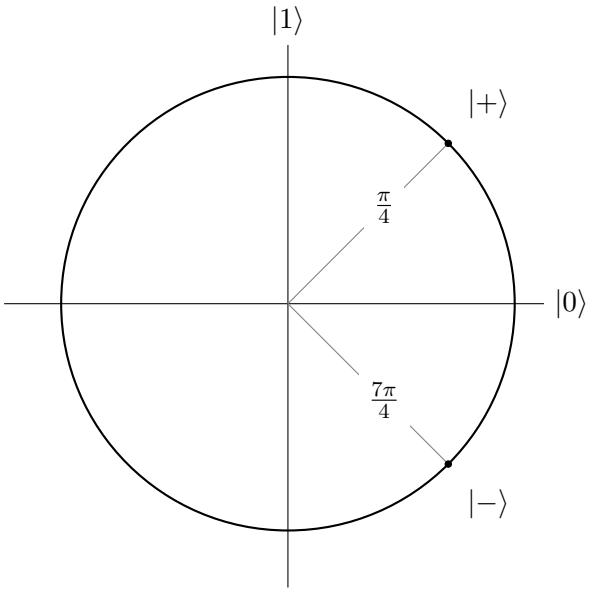
- Particle spins: $|\text{up}\rangle, |\text{down}\rangle$
- Trapped atom and ion energies: $|\text{low}\rangle, |\text{high}\rangle$
- Photon polarizations: $|\text{vertical}\rangle, |\text{horizontal}\rangle$
- Super-conducting circuit current direction: $|\text{clockwise}\rangle, |\text{counter} - \text{clockwise}\rangle$

We will learn more about the physical realizations of quantum computing through our group literature research projects!

In addition to the $|0\rangle, |1\rangle$ bases, we can also express qubits in the $|+\rangle, |-\rangle$ basis, as

$$|+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}},\ |-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}} \tag{3}$$

which we can visualize as



If we are maximally certain in the $|0\rangle, |1\rangle$ basis, we are maximally uncertain in the $|+\rangle, |-\rangle$ basis, and vice versa.

2.2 Multiple qubits

We can also consider systems of multiple systems, of the form (for a 2-qubit state)

$$|\psi\rangle = \alpha_1 |00\rangle + \alpha_2 |01\rangle + \alpha_3 |10\rangle + \alpha_4 |11\rangle \tag{4}$$

where $|00\rangle = |0\rangle \otimes |0\rangle$ (both qubits are in $|0\rangle$), $|01\rangle = |0\rangle \otimes |1\rangle$ (the first qubit is in $|0\rangle$ and the second qubit is in $|1\rangle$, and so on.

We can express the states in column notation as

$$|00\rangle = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, |01\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, |10\rangle = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, |11\rangle = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{5}$$

and thus our state is

$$|\psi\rangle = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} \tag{6}$$

For ease of notation, we have a tensor product of n qubits each in a state $|0\rangle$, we'll write this as

$$|0\rangle \otimes |0\rangle \otimes ..(n \text{ total times}) = |0000...(n \text{ total times})\rangle = |0\rangle^{\otimes n} \tag{7}$$

Similarly, we can write states as

$$|001110\rangle = |0\rangle^{\otimes 2} |1\rangle^{\otimes 3} |0\rangle \tag{8}$$

3 Building block: quantum gates

Let's now learn about some quantum gates that we will see throughout various quantum computing algorithms.

3.1 Hadamard gate

A Hadamard gate maps a qubit between the $|0\rangle, |1\rangle$ basis and the $|+\rangle, |-\rangle$ basis.

We express it as

$$\mathrm{H} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{9}$$

Operating on the $|0\rangle, |1\rangle$ basis, we have

$$\mathrm{H}|0\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = |+\rangle \tag{10}$$

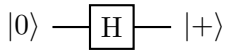
$$\mathrm{H}|1\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix} = |-\rangle \tag{11}$$

Note that the Hadamard gate is its own inverse.

On paper: Convince yourself that

$$\mathrm{H}|+\rangle = |0\rangle, \mathrm{H}|-\rangle = |1\rangle \tag{12}$$

We write the Hadamard gate in circuit notation as



(13)

Now, if we want to Hadamard a state composed of n qubits, we apply a Hadamard gate to each of the qubits as

$$\mathrm{H}^{\otimes n} = \mathrm{H} \otimes \mathrm{H} \otimes \mathrm{H} \otimes \ldots \text{n total} \tag{14}$$

Think, pair, share: Suppose we have n qubits, each in the state $|0\rangle$, so that we have a state $|\psi\rangle = |0\rangle^{\otimes n}$. What would we get by applying a Hadamard gate to each of the qubits, as $\mathrm{H}^{\otimes n}$? Why could this be useful?

The Hadamard gate maps n $|0\rangle$ qubits into a superposition of 2^n orthonormal states in the $|0\rangle, |1\rangle$ basis with equal weight.

The Hadamard gate, as we shall see throughout this course, is used in many quantum algorithms, including the Deutsch-Jozsa algorithm, Grover's algorithm (for quantum search), and Shor's algorithm (for factoring).

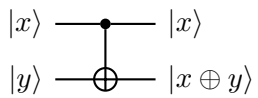
3.2 cNOT gate

Controlled NOT gate: Opertates on 2 qubit state, flips the second qubit if and only if the first qubit is $|1\rangle$. Recalling our convention for 2-qubit systems in Eq. (5), we represent this by the matrix

$$\mathrm{cNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{15}$$

On paper: Convince yourself that this matrix has the desired behavior - what does it do to the states $|00\rangle$, $|01\rangle$, $|10\rangle$ and $|11\rangle$?

We can think about this in terms of the exclusive or (XOR) operation, as $|x, y\rangle \to |x, x \oplus y\rangle$, where $\oplus$ represents XOR, or addition modulo 2. We represent this by a circuit as



(16)

where $\bullet$ represents the "control", and $\oplus$ represents XOR.

While we will be using the cNOT gate for quantum oracles, it can also be used to construct a Bell state!

4 Building block: quantum oracles

Now let's talk about quantum oracles, an important construct in both quantum algorithms and quantum complexity theory. At its heart, a quantum oracle is a black-box function of the form

$$\text{Quantum Oracle: } f : \{0,1\}^n \to \{0,1\} \tag{17}$$

mapping a state of n bits to a boolean result (one number, either 0 or 1).

When implementing various quantum algorithms, we'll want to learn some property of f by only evaluating f on various inputs, without actually looking at the internals of f. In particular, we'll want to harness the power of quantum computing to minimize the number of queries we make to f to learn about some property of it.

Let's think about how to implement the quantum oracle as a quantum mechanical operation. Let's refer to the oracle as the operator U_f.

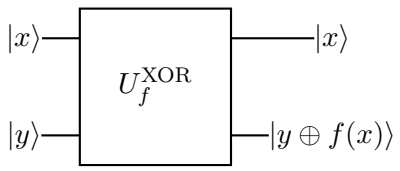
Think, pair, share: Suppose we consider implementing the quantum oracle through something simple like $U_f |x\rangle = |f(x)\rangle$ where $|x\rangle$ is an n-qubit state. What could go wrong here? What could this break?

In order to ensure that we have a unitary, reversible operation, we can use a one-qubit *answer bit* or *target register* $|y\rangle$, creating a map such as

$$U_f^{\text{XOR}} |x, y\rangle = |x, y \oplus f(x)\rangle \tag{18}$$

where $|x\rangle$ is an n-qubit state, and $\oplus$ is the XOR operation (using a cNOT gate), which also corresponds to addition modulo 2.

Expressing this as a quantum circuit, we have



(19)

Question: What is $U_f^{\text{XOR}} U_f^{\text{XOR}} |x, y\rangle$? What does this tell us about U_f^{XOR}?

A more convenient oracle to use sometimes is the *phase oracle*, which maps

$$U_f^{\text{Phase}} |x, b\rangle = (-1)^{f(x) \cdot b} |x, b\rangle \tag{20}$$

Where b is a target register. This, as we shall see, is equivalent to the XOR Oracle, and hence has the same properties. Suppose for the XOR Oracle we prepare the target register as $|b\rangle = |-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}$, calling the XOR oracle on

$$U_f^{\text{XOR}} \left(\frac{|x, 0\rangle - |x, 1\rangle}{\sqrt{2}} \right) \tag{21}$$

Question: What does the XOR oracle in Eq. (21) when $f(x) = 0$? When $f(x) = 1$? Does this have the behavior of the phase oracle $|x, -\rangle \to (-1)^{f(x)} |x, -\rangle$?

For homework, we'll show the reverse, going from the phase oracle to the XOR oracle.

5 The Deutsch-Jozsa algorithm

Now we'll use our new quantum computing building blocks to explore the Deustch-Jozsa algorithm, which is a quantum algorithm that is much faster than any corresponding deterministic classical algorithm. Though it doesn't have the same level of real-world application such as quantum search and factoring algorithms, it is a nice example of illustrating the power of quantum computing over classical computing.

The problem: Suppose we are given a black-box oracle that implements a function

$$f : \{0, 1\}^n \to \{0, 1\} \tag{22}$$

operating on n bits and returning either 0 or 1. We are *guaranteed* that f is either:

- **Constant**: returns 0 for all inputs or 1 for all inputs
- **Balanced**: returns 0 for half of the inputs and 1 for half of the inputs

Given a black-box oracle f, determine whether f is balanced or constant.

5.1 The classical case

First, let's consider implementing the problem statement in a *classical* way.

Finger Poll: How many possible inputs does f take?

1. n

2. $2n$
3. 2^n

Now let's consider the best and worst-case behavior for answering the problem in a classical way.

Question: What is the best-case scenario to determine that f is balanced?

Think, pair, share: What is the number of inputs (and hence queries to f) we'll need to determine that f is constant?

Thus, we see that in the classical case, we'll need a large number of queries, $2^{n-1}+1$, to determine whether f is balanced or constant.

5.2 The quantum case

Now let's see how much better a quantum algorithm could do. First, let's write each of the possible 2^n inputs to f as quantum state

$$|x\rangle, x \in \{0,1\}^n$$

We can write the 2^n states like

$$|0\rangle^{\otimes n}$$
$$|0\rangle^{\otimes(n-1)} |1\rangle$$
$$|0\rangle^{\otimes(n-2)} |1\rangle |0\rangle$$
$$|0\rangle^{\otimes(n-2)} |1\rangle^{\otimes 2}$$
$$\dots$$
$$|1\rangle^{\otimes(n-1)} |0\rangle$$
$$|1\rangle^{\otimes n}$$

On paper: Write out all of the possible states for $n = 3$

Think, pair, share: Ultimately, we want to query $f(x)$ with all possible $|x\rangle$, $x \in \{0,1\}^n$. How can we prepare a quantum state that is a superposition of all $|x\rangle$? Think about one of the building blocks we used earlier :)

Step 0: Let's begin with n qubits in the state $|0\rangle$

$$|\psi_0\rangle = |0\rangle^{\otimes n} \tag{23}$$

Step 1: Now let's Hadamard each of the bits, giving us

$$|\psi_1\rangle = \mathrm{H}^{\otimes n} |\psi_0\rangle = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle \tag{24}$$

By using the Hadamard gate, we're set with all of the possible inputs to the quantum oracle that we want to consider.

For homework, we will derive the more general expression that $\mathrm{H}^{\otimes n}$ operating on a state $|x\rangle$ of n qubits takes the form

$$\mathrm{H}^{\otimes n} |x\rangle = \sum_{z\in\{0,1\}^n} \frac{1}{\sqrt{2^n}} (-1)^{x\cdot z} |z\rangle \tag{25}$$

where $|z\rangle$ is an n-qubit state and $x \cdot z$ is the sum of the bitwise product $x \cdot z = x_0 z_0 \oplus x_1 z_1 \oplus \ldots x_{n-1} z_{n-1}$ where $\oplus$ is addition modulo 2.

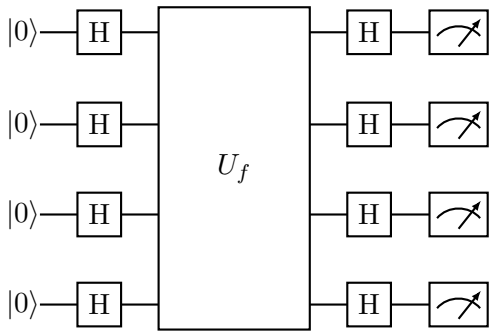
Step 2: Let's now call the quantum oracle U_f, in the form of a phase oracle. Recalling that the phase oracle maps $|x\rangle \to (-1)^{f(x)} |x\rangle$, we obtain the state

$$|\psi_2\rangle = U_f |\psi_1\rangle = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} (-1)^{f(x)} |x\rangle \tag{26}$$

Step 3: Hadamard all of the qubits in $|\psi_2\rangle$ by applying $\mathrm{H}^{\otimes n}$. Recall the expression for $\mathrm{H}^{\otimes n}$ in Eq. (25). Combining this with Eq. (26) gives

$$\begin{aligned} |\psi_3\rangle = \mathrm{H}^{\otimes n} |\psi_2\rangle &= \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} (-1)^{f(x)} \frac{1}{\sqrt{2^n}} \sum_{y\in\{0,1\}^n} (-1)^{x\cdot y} |y\rangle \\ &= \frac{1}{2^n} \sum_{x\in\{0,1\}^n} \sum_{y\in\{0,1\}^n} (-1)^{f(x)} (-1)^{x\cdot y} |y\rangle \end{aligned} \tag{27}$$

Our quantum circuit thus looks like (for 4 qubits, for example)



(28)

Step 4 (last step): measure our qubits.

Question: Given the state $|\psi_3\rangle$, what's the probability of measuring $|y\rangle = |0\rangle^{\otimes n}$?

Think, pair, share:

Now considering this probability,

$$P(|0\rangle^{\otimes n}) = \left\| \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{f(x)} \right\|^2 \tag{29}$$

what is $P(|0\rangle^{\otimes n})$ going to be if f is constant? What if f is balanced?

We see that $P(|0\rangle^{\otimes n}) = 1$ if f is constant, and 0 if f is balanced. In other words, we are guaranteed to measure the state $|0\rangle^{\otimes n}$ is f is constant, and if f is balanced, we will measure any other state. Thus, to solve the original problem, all we need to do is perform a measurement!

Finger Poll: Recall that we need to make $2^{n-1} + 1$ queries to f to solve the problem in the classical case. How many queries did we make to f in the quantum case?

1. n
2. $2^{n-1} + 1$
3. 1

and that's the power of quantum computing!

Two minute essay:

1. What do I know now that I didn't before?
2. What is confusing?
3. What do I want to know more about?

6 For next time

Continue to work on your group presentations on physical realizations of quantum computers (Nielsen and Chuang, Chapter 7)

Next class, we're going to take a look at simulating the behavior of a quantum computer using the python package qiskit. If you would like, you can take a look at one implementation of the Deutsch-Jozsa algorithm in qiskit at https://qiskit.org/textbook/ch-algorithms/deutsch-jozsa.html

Written exercises

Exercise 1. Derive the expression in Eq. (25). Note that $H^{\otimes n} = H \otimes H^{\otimes(n-1)}$

Exercise 2. Recall our study of the Bloch sphere to express qubits. What does the Hadamard gate do on the Bloch sphere?

Exercise 3. Practice with gates: What control and target states $|\psi_A\rangle$ and $|\psi_B\rangle$ would we pass to a cNOT gate in order to obtain a Bell state?

Exercise 4. Practice with oracles: Suppose we have a state $|\psi\rangle = \alpha\,|00\rangle + \beta\,|01\rangle + \delta\,|10\rangle + \gamma\,|11\rangle$. What is $U_f^{\text{XOR}}\,|\psi\rangle$?

Exercise 5. In class, we showed that we can map the XOR oracle to the phase oracle by setting the target bit as $|y\rangle = |-\rangle$. Show that we can map the phase oracle to the XOR oracle.

Exercise 6. Improving on the classical algorithm: Let's now consider the probabilistic classical case of the Deustch-Jozsa algorithm, which may help us improve on the $2^{n-1} + 1$ queries that we have to make in the classical case. Suppose that we want to know whether $f(x)$ is constant or balanced not with *certainty*, but with some probability of error $\epsilon < \frac{1}{2}$. How may queries would we need to make to $f(x)$?